Recuperação de Informação / Information Retrieval

2019/2020 MIECT/MEI, DETI, UA

Assignment 1

Submission deadline: **16 October 2019**

For this assignment you will create a simple document indexer, consisting of a corpus reader / document processor, tokenizer, and indexer.

For this assignment you will use the corpus available here: https://bit.ly/2lUmvLe

1. Create a corpus reader class that iterates over the collection (corpus) of document and returns, in turn, the contents of each document. For this assignment consider only the identifier (PMID) and title (TI) fields.

2. Create different tokenizer implementations.

   2.1. A simple tokenizer that replaces all non-alphabetic characters by a space, lowercases tokens, splits on whitespace, and ignores all tokens with less than 3 characters.

   2.2. An improved tokenizer that incorporates your own tokenization decisions (e.g. how to deal with digits and characters such as ’, -, @, etc).
   Integrate the Porter stemmer (http://snowball.tartarus.org/download.html) and a stopword filter. Use this list as default: https://bit.ly/2kKBCqt

3. Create an indexer class to index the corpus using a suitable data structure defined by you. Index the corpus using the simple tokenizer from 2.1. and that from 2.2. (with stemming and stopword filtering active) and write the resulting indexes to file using the following format (one term per line):

```
aaaaa,doc id:term freq,doc id:term freq,…
aaaab,doc id:term freq,doc id:term freq,…
aaaac,doc id:term freq,doc id:term freq,…
(…)
```

4. For each tokenizer above, answer the following questions:
   a) What is the total indexing time and final index size on disk?
   b) What is your vocabulary size?
   c) List the ten first terms (in alphabetic order) that appear in only one document (document frequency = 1).
   d) List the ten terms with highest document frequency.

Grading will be based on:

- Complete and correct implementation;
- Modelling, class diagram, code structure, organization and readability, correct use of data structures, submitted results, and report.

Note:

Your implementation should be modular and easily extended/adapted to other corpora structures.

See suggestions and submission instructions in the next page.

Suggestions:

- Write **modular** code
- Favour **efficient** data structures
- Use **parameters**, preferably through the command line
- Add **comments** to your code

Submission instructions:

- To manage your project please use **Maven**
- Include a small **Report** including:
    - Your project's **class diagram**
    - A short description of each class and main methods, and a high-level (but sufficiently detailed) description of the overall processing pipeline (data flow). You may add a block diagram to support this description.
    - If necessary, instructions on how to run your code, including any parameters that should be used/changed
    - Answers to the questions above
- Make sure you **include your name and student number** in the code and in the report
- Make sure all your programs compile and run correctly
- Submit your assignment by the due date using Moodle